CLAIMS

What is claimed is:

1. A system that facilitates asynchronous programming, comprising:

a contract component that facilitates stating a behavioral contract on an asynchronous service interface between a client interface and a service interface; and

a conformance checking component that checks that at least one of the client interface and the service interface conform to the contract defined therebetween.

2. The system of claim 1, the contract defines an order in which clients of the service interface are invoked.

3. The system of claim 2, the checking component checks that the clients handle both normal and exceptional results.

4. The system of claim 1, at least one of the asynchronous service interface and the contract are defined according to an objected oriented program language.

5. The system of claim 1, the checking by the checking component is modular wherein the contract is specified on an interface boundary of the client interface and the service interface and, each of the client interface and the service interface is checked independently, and each method is checked separately.

6. The system of claim 1, the checking component further comprises a model extraction component that automatically extracts a behavior model that is fed to a model checker component.

7. The system of claim 6, the extracted model is sound in the presence of aliasing.

8. The system of claim 1, further comprising a construct in the form of an asynchronous call that returns a future-like handle to an eventual result of the call.

9. The system of claim 1, further comprising a construct in the form of a synch statement.

10. The system of claim 1, further comprising a construct that states the behavioral contract on the service interface.

11. The system of claim 1, further comprising a construct that is a transaction directive that delineates the scope of conversations by clients.

12. The system of claim 1, further comprising a construct that is a tracked type qualifier used by the checking component.

13. The system of claim 1, further comprising a procedure, which procedure is a method that is called at least one of synchronously and asynchronously.

14. The system of claim 13, the asynchronous call executes concurrently with its caller by completing immediately, while the method that is invoked executes.

15. The system of claim 14, the asynchronous call completes by returning a value.

16. The system of claim 15, the value is used explicitly to synchronize the asynchronous call with the value.

17. The system of claim 15, the value is a first-class type that can be at least one of stored, passed as an argument, and returned as a result.

18. The system of claim 17, the result returned is explicitly retrieved through a join statement.

19. The system of claim 15, the value is an object in one of three states that include an undefined state, normal state, and exceptional state.

20. The system of claim 1, the contract is a source-level contract that facilitates statically detecting asynchronous message passing errors.

21. The system of claim 1, the contract expresses stateful protocols that govern interactions between a client associated with the client interface and a service associated with the service interface.

22. The system of claim 1, the service interface is associated with a service contract whose language models the effects of multiple clients that concurrently access the service interface.

23. The system of claim 1, the checking component checks that services and clients obey the contract associated with the service interface.

24. The system of claim 1, the checking component uses a transaction directive to delimit a scope of an instance of a concurrent interaction between a client and a set of service instances.

25. The system of claim 24, the transaction directive specifies a correlation variable that uniquely identifies the instance of the concurrent interaction between the client and the set of service instances.

26. The system of claim 1, the checking component is modular in that to check for the client conformance and the service conformance, only the associated contract needs to be referenced.

27. The system of claim 1, the checking component considers a method as a function of an object to which it belongs.

28. The system of claim 1, the checking component relates a contract state, a client state, and a service state.

29. The system of claim 28, the checking component further comprises an extraction component that extracts a client model, service model, and interface model in order to relate the contract state, client state, and service state.

30. The system of claim 28, the checking component further comprises an extraction component that uses a region to model relevant data, which relevant data is that data which is directly relevant to the contract state.

31. The system of claim 30, the region is polymorphic.

32. The system of claim 30, the region is partial such that only relevant data is directly assigned to a region type.

33. The system of claim 1, the checking component further comprises an extraction component that facilitates model reduction by utilizing only necessary statements.

34. The system of claim 1, the checking component further comprises an extraction component that utilizes region-and-effect analysis to extract a model.

35. The system of claim 1, the checking component further comprises an extraction component that utilizes type-and-effect inference and source-level tracked types.

36. The system of claim 1, the contract is programmer-specified to separately check client and server code.

37. A computer according to the system of claim 1.

38. A system that facilitates asynchronous programming, comprising:

a client interface and a service interface;

an extraction component that automatically extracts a client model from the client interface and a service model from the service interface;

a relator component that creates a behavioral relationship between the client model and the service model; and

a conformance checking component that checks that the client interface obeys the behavioral relationship, and the service interface properly implements the behavioral relationship.

39. The system of claim 38, the behavioral relationship expressed in terms of at least one of a future, a join on the future, and asynchronous function calls.

40. The system of claim 38, the behavioral relationship is a contract the terms of which are enforced by the checking component.

41. The system of claim 38, the checking component checks that clients of the client interface are invoked in the proper order and that the clients handle both normal and exceptional results.

42. The system of claim 38, the extraction component creates the client model that defines relevant data for interaction with the service interface.

43. The system of claim 38, the extraction component creates the service model that defines relevant data for interaction with the client interface.

44. The system of claim 38, the relator component checks the client model and the service model such that contract descriptions of the behavioral relationship are defined between the client model and the service model.

45. A method of asynchronous programming, comprising:

receiving a client interface and a service interface;

creating a behavioral relationship between the client interface and the service interface; and

checking to ensure that the client interface obeys the behavioral relationship and the service interface properly implements the behavioral relationship.

46. The method of claim 45, the behavioral relationship is expressed in terms of at least one of a future, a join on the future, and asynchronous function calls.

47. The method of claim 45, the behavioral relationship is a contract the terms of which are enforced by a modular checking component.

48. The method of claim 47, the modular checking component checks that clients of the client interface are invoked in the proper order and that the clients handle both normal and exceptional results.

49. The method of claim 45, the further comprising extracting with a model extraction component at least one of a client model for the client interface and a service model for the service interface.

50. The method of claim 49, the model extraction component creating the client model that defines relevant data for interaction with the service interface.

51. The method of claim 49, the model extraction component creates the service model that defines relevant data for interaction with the client interface.

52. The method of claim 49, further comprising performing model checking of the client model and the service model with a model checker such that contract descriptions are defined between the client model and the service model.

53. The method of claim 45, further comprising automatically extracting the behavioral relationship

54. A method of asynchronous programming, comprising:

receiving a client interface and a service interface;

automatically extracting a client model from the client interface and a service model from the service interface;

creating a behavioral relationship between the client model and the service model; and

enforcing the behavioral relationship between the client interface and the service interface.

55. The method of claim 54, the behavioral relationship is a contract expressed by constructs in terms of at least one of a future, a join on the future, and asynchronous function calls, and the terms of which are enforced by a modular checking component.

56. The method of claim 54, further comprising checking that clients of the client interface are invoked in the proper order and that the clients handle both normal and exceptional results.

57. The method of claim 54, further comprising:

extracting with a model extraction component a client model from the client interface, the client model defines relevant data for interaction with the service interface for the client interface; and

extracting with the model extraction component a service model from the service interface, the service model defines relevant data for interaction with the client interface.

58. A computer software product that embodies program instructions for execution by a processing unit, the processing unit retrieving the instructions from a computer-readable memory to facilitate concurrent execution of the instructions, the product comprising:

a contract component that creates a contract state between a client state and a service state;

an extraction component that automatically extracts a client model from the client state, a service model from the service state, and an interface model from the contract state; and

a conformance checking component that checks that the client model obeys the interface model, and the service model properly implements the contract model.

59. The product of claim 58, the client model, service model, and interface model contain only respective communication actions and operations that are relevant for maintaining the state of a communication protocol.

60. The product of claim 59, the relevant operations are represented by the use of regions.

61. The product of claim 58, the interface model includes one or more effect processes for each method of the interface state.

62. The product of claim 61, the effect process models a method of the interface state according to an effect a call has on the contract state, a future created by an asynchronous call to the method, and futures passed into the method.

63. A system of asynchronous programming, comprising:

means for receiving a client interface and a service interface;

means for automatically extracting a client model from the client interface and a service model from the service interface;

means for creating a behavioral relationship between the client model and the service model; and

means for checking to ensure that the client interface obeys the behavioral relationship and the service interface properly implements the behavioral relationship.